Distributed Systems : Assignment 1

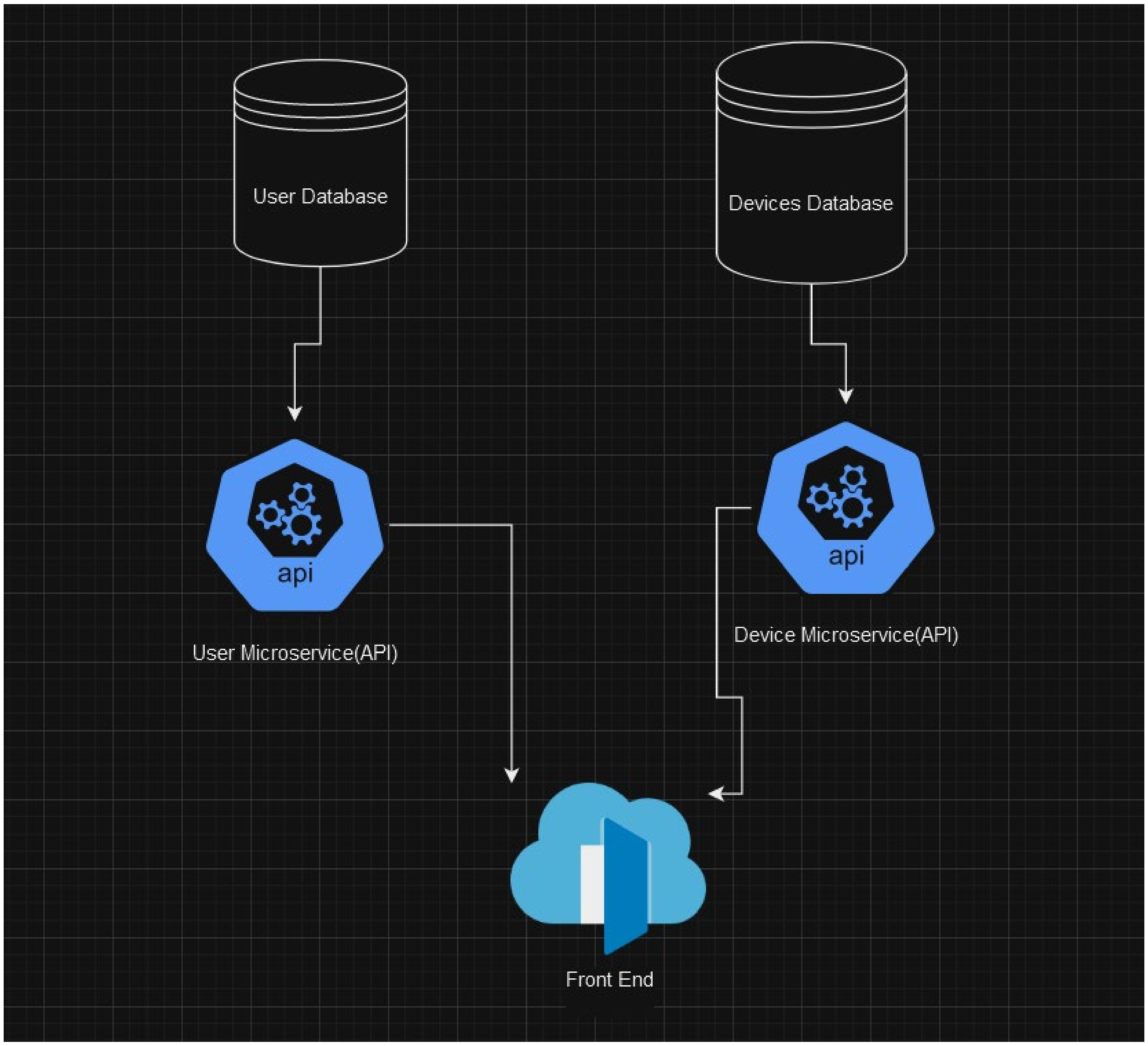
Introduction

The goal of the project was to develop an Energy Management System. It consists of a front-end that should communicate with two other microservices: One designed for user management and one for the management of the energy metering devices associated with each user. The system has 2 user types: administrator and user/client. The administrator has higher rights, being able to perform CRUD operations on both users and devices and perform the mapping between them. The user can perform CRUD operations on their own devices and their own account.

The back-ends were developed using Java and its Spring Boot framework ,exposing a REST API to interact with the database via requests. The databases for both of them were implemented using PostgreSQL. These requests are made with the aid of a frontend developed using JavaScript and React+Vite. All these components were deployed in their own unique container, and are interconnected, being deployed on the same network using docker-compose.

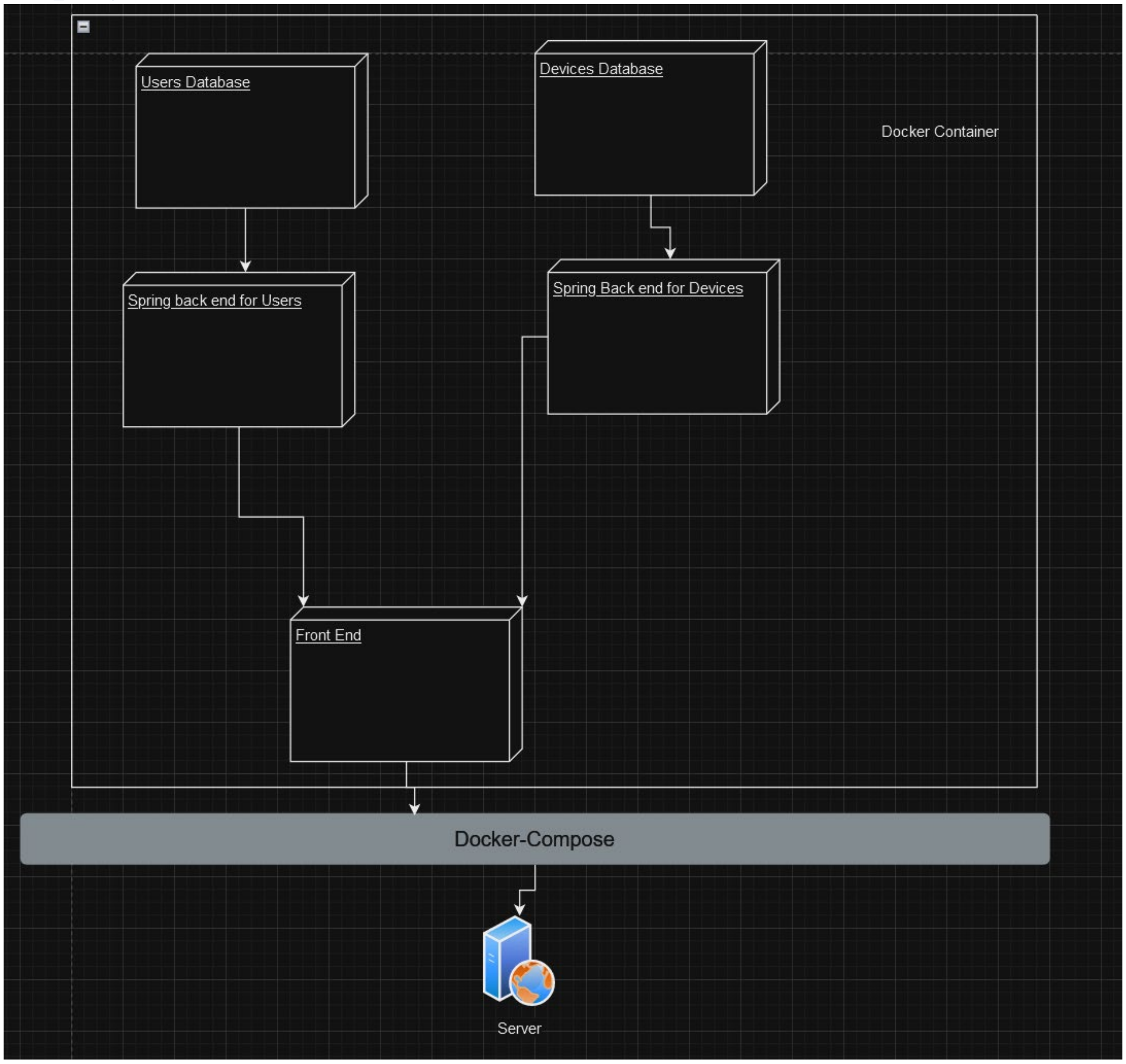
Architecture of the application

The back-end was split into 2 microservices: Device and User ,each with their own separate database. The user database contains logic for registering and authenticating users in the system using JWT. And all other requests require a JWT token in order to be performed. The 2 services are also synchronized, as in when a new user is created, an entry in the available user IDs table in the device database is also created.



Since I’ve used Spring Data JPA, the database will be automatically generated when the application is first run, and will then be only updated afterwards, and data will persist. There are some sql scripts in the microservices’ directories, should you want to manually run them in psql inside the docker containers.

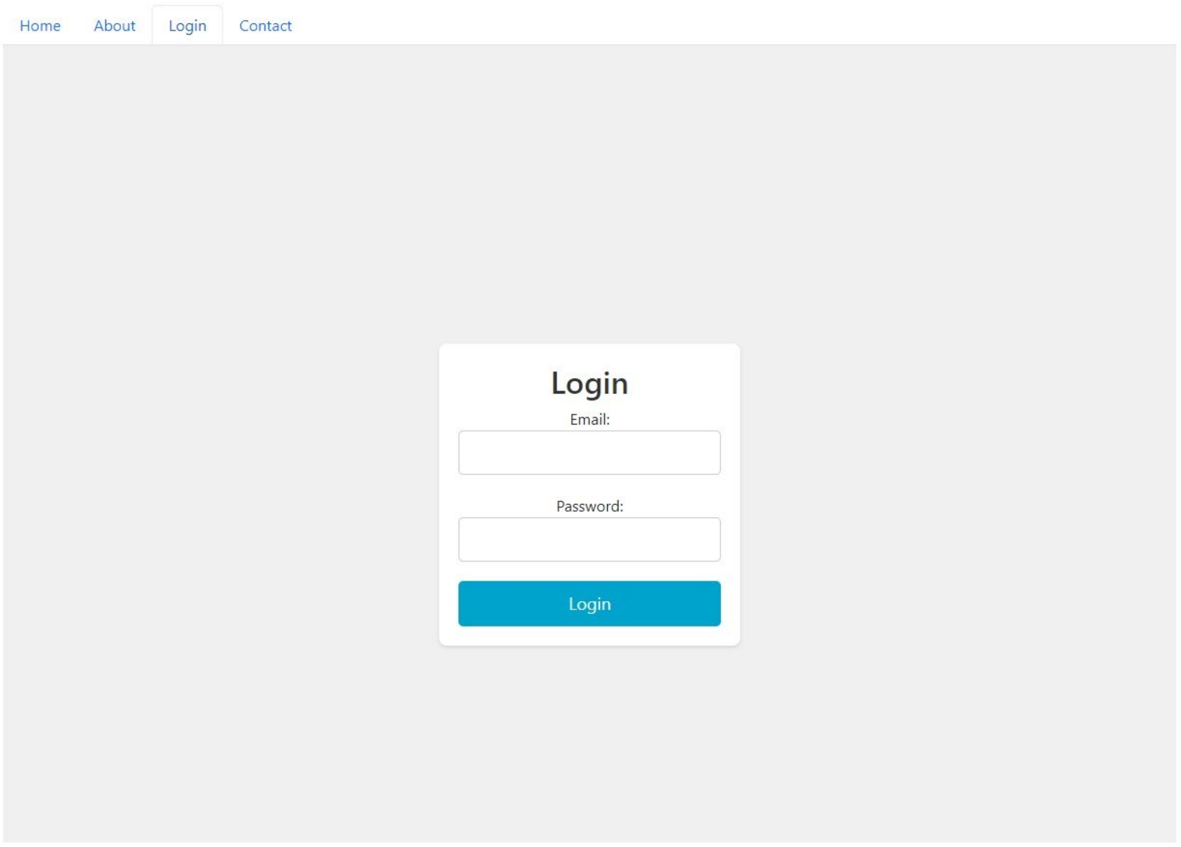
Deployment :



In order to run the application, I had to create the JAR files for the Java applications and the application properties file for the React frontend, then deploy them in images from inside docker. The final result contained a large container built using docker-compose, which contained the 5 smaller containers for every component of the whole application.

Functionality examples:

Login Page:

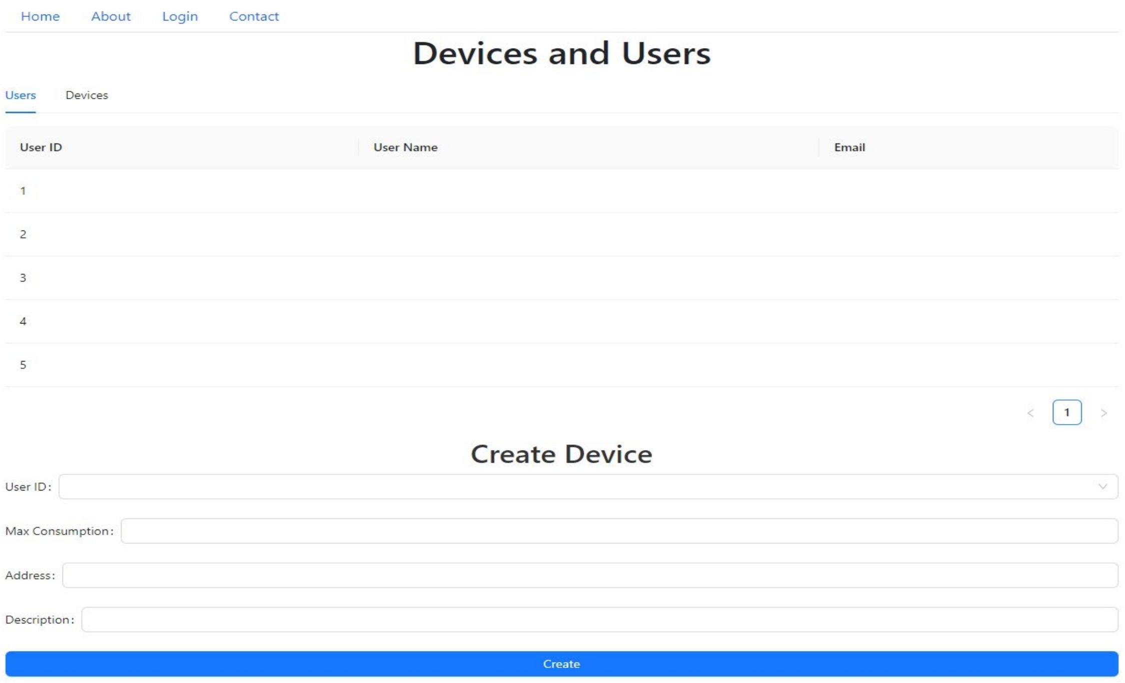


Login

Email:

Password:

Administrator Dashboard to see user Id’s and device creation form:



Devices and Users

User ID	User Name	Email
1		
2		
3		
4		
5		

Create Device

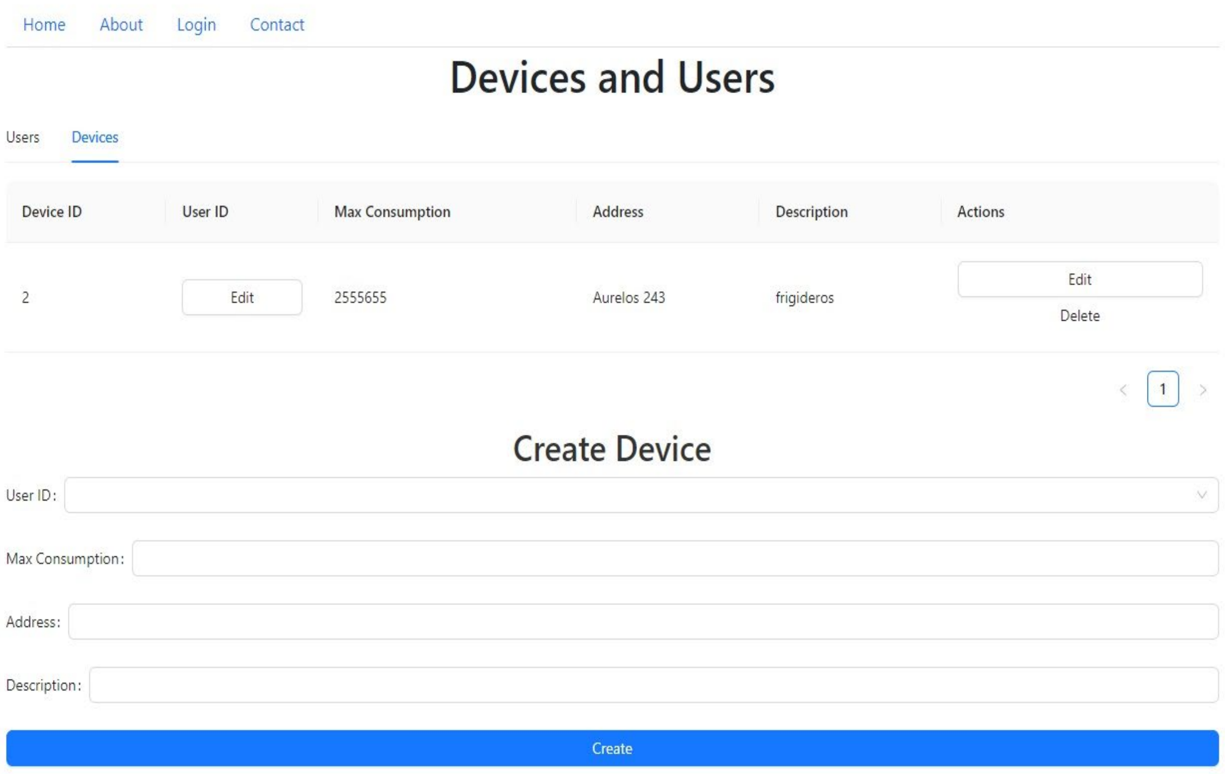
User ID:

Max Consumption:

Address:

Description:

Administrator dashboard for device management:



Devices and Users

Device ID	User ID	Max Consumption	Address	Description	Actions
2	Edit	2555655	Aurelos 243	frigideros	Edit Delete

Create Device

User ID:	
Max Consumption:	
Address:	
Description:	